L Number	Hits	Search Text	DB	Time stamp
-	13697	tone with (detect$4 or decod$3)	USPAT	2003/07/15 09:56
-	212278	(activat$4 or deactivat$4 or de-activat$4) same (processor or control$3)	USPAT	2003/07/12 14:15
-	4009	(tone with (detect$4 or decod$3)) and ((activat$4 or deactivat$4 or de-activat$4) same (processor or control$3))	USPAT	2003/07/12 14:01
-	841	(tone with (detect$4 or decod$3)) same ((activat$4 or deactivat$4 or de-activat$4) same (processor or control$3))	USPAT	2003/07/12 14:02
-	8746	(line adj card) or (subscriber with circuit)	USPAT	2003/07/12 14:10
-	26	((tone with (detect$4 or decod$3)) same ((activat$4 or deactivat$4 or de-activat$4) same (processor or control$3))) same ((line adj card) or (subscriber with circuit))	USPAT	2003/07/12 14:13
-	255105	(plurality with (processor or control$4 or cpu or microprocess$3))	USPAT	2003/07/12 14:26
-	593517	(activat$4 or deactivat$4 or de-activat$4)	USPAT	2003/07/12 14:26
-	12034	((plurality with (processor or control$4 or cpu or microprocess$3))) same ((activat$4 or deactivat$4 or de-activat$4))	USPAT	2003/07/12 14:15
-	47	(tone with (detect$4 or decod$3)) same (((plurality with (processor or control$4 or cpu or microprocess$3))) same ((activat$4 or deactivat$4 or de-activat$4)))	USPAT	2003/07/12 14:16
-	67349	(plurality with (processor or control$4 or cpu or microprocess$3))	EPO; JPO; DERWENT; IBM_TDB	2003/07/12 14:26
-	329940	(activat$4 or deactivat$4 or de-activat$4)	EPO; JPO; DERWENT; IBM_TDB	2003/07/12 14:26
-	11024	tone with (detect$4 or decod$3)	EPO; JPO; DERWENT; IBM_TDB	2003/07/12 14:26
-	4	((plurality with (processor or control$4 or cpu or microprocess$3))) and ((activat$4 or deactivat$4 or de-activat$4)) and (tone with (detect$4 or decod$3))	EPO; JPO; DERWENT; IBM_TDB	2003/07/12 14:27
-	0	(fir adj filter) same (time adj dmain)	USPAT	2003/07/15 09:57
-	280	((fir or (finite adj impulse adj response)) adj filter) same (time adj domain)	USPAT	2003/07/15 10:00
-	14	(((fir or (finite adj impulse adj response)) adj filter) same (time adj domain)) and ((head adj relate$1 adj transfer$4 adj function) or hrtf)	USPAT	2003/07/15 10:04

transfer function

transfer function: **1.** A mathematical statement that describes the transfer characteristics of a system, subsystem, or equipment. **2.** The relationship between the input and the output of a system, subsystem, or equipment in terms of the transfer characteristics. *Note 1:* When the transfer function operates on the input, the output is obtained. Given any two of these three entities, the third can be obtained. *Note 2:* Examples of simple transfer functions are voltage gains, reflection coefficients, transmission coefficients, and efficiency ratios. An example of a complex transfer function is envelope delay distortion. *Note 3:* For a negative feedback circuit, the transfer function, T, is given by

$$T = \frac{e_0}{e_i} = \frac{G}{1 + GH},$$

where e_0 is the output, e_i is the input, G is the forward gain, and H is the backward gain, *i.e.,* the fraction of the output that is fed back and combined with the input in a subtracter. **3** . Of an optical fiber, the complex mathematical function that expresses the ratio of the variation, as a function of modulation frequency, of the instantaneous power of the optical signal at the output of the fiber, to the instantaneous power of the optical signal that is launched into the fiber. *Note:* The optical detectors used in communication applications are square-law devices. Their output current is proportional to the input optical power. Because electrical power is proportional to current, when the optical power input drops by one-half (3 dB), the electrical power at the output of the detector drops by three-quarters (6 dB). [FAA]

Infinite Impulse Response filters

An Infinite Impulse Response (IIR) filter produces an output, $y(n)$, that is the weighted sum of the current and past inputs, $x(n)$, and past outputs.

The Linear Predictive model is a specical case of an IIR filter and shown in figure 7.

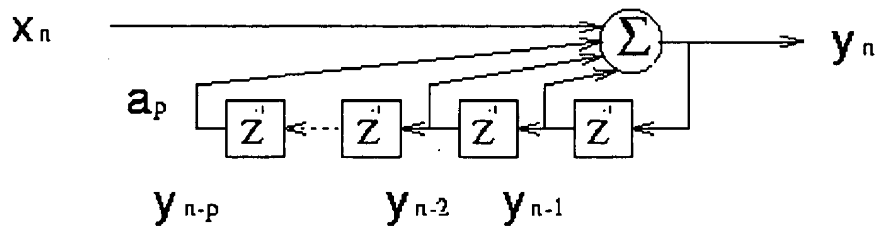

Figure 7: An IIR filter

The general IIR filter (figure 8) is given by:

$$y_n = \sum_{i=1}^{p} a_i y_{n-i} + \sum_{j=0}^{q} b_j x_{n-j} \tag{11}$$

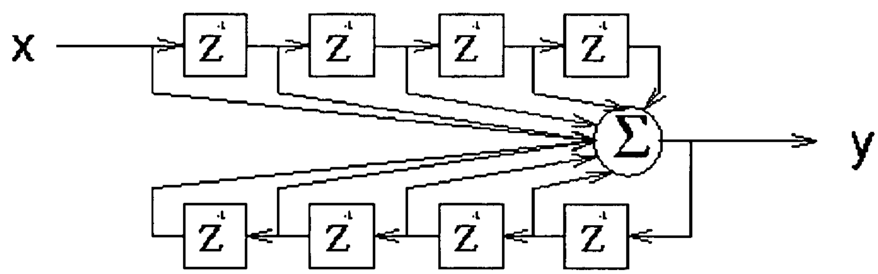

Figure 8: The general linear filter

If $p = 0$ then the system represents a finite impulse response (FIR) filter. If p is not zero, then the system is an infinite impulse response (IIR) filter.

An example is the two pole resonator with center frequency ω and bandwidth related to r is:

$$y_n = 2r\cos(\omega T)y_{n-1} - r^2 y_{n-2} + x_n - \cos(\omega T)x_{n-1} \tag{12}$$

Common types of IIR filter:

Type	Characteristics
Butterworth	maximally flat amplitude
Bessel	maximally flat group delay
Chebyshev	equiripple in passband or stop-band
Elliptic	equiripple in passband and stop-band

For more details see [4].

Finite Impulse Response filters

A Finite Impulse Response (FIR) filter produces an output, $y(n)$, that is the weighted sum of the current and past inputs, $x(n)$.

$$y_n = b_0 x_n + b_1 x_{n-1} + b_2 x_{n-2} + \ldots + b_q x_{n-q} \quad (1)$$
$$= \sum_{j=0}^{q} b_j x_{n-j} \quad (2)$$

This is shown in figure 4 with z^{-1} representing a unit delay.

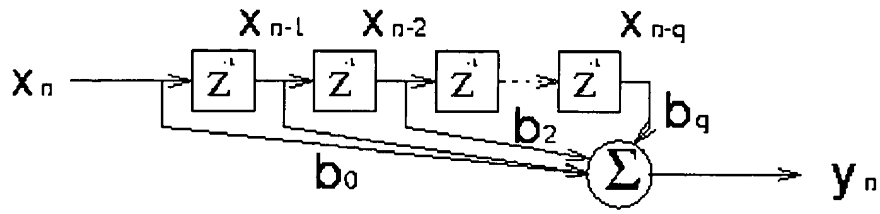

Figure 4: A FIR filter

Consider supplying this filter with a sine wave, $x_n = \sin(\omega n T)$:

$$y_n = \sum_{j=0}^{q} b_j \sin(\omega(n-j)T) \quad (3)$$

Using the identity $\sin(\theta + \phi) = \sin\theta\cos\phi + \cos\theta\sin\phi$:

$$y_n = \sum_{j=0}^{q} b_j \left(\sin(\omega n T)\cos(-\omega j T) + \cos(\omega n T)\sin(-\omega j T)\right) \quad (4)$$
$$= \left(\sum_{j=0}^{q} b_j \cos(-\omega j T)\right)\sin(\omega n T) + \left(\sum_{j=0}^{q} b_j \sin(-\omega j T)\right)\cos(\omega n T) \quad (5)$$

The terms in parantheses are independent of time and hence the output is a sinusoid with amplitude:

$$\sqrt{\left(\left(\sum_{j=0}^{q} b_j \cos(-\omega j T)\right)^2 + \left(\sum_{j=0}^{q} b_j \sin(-\omega j T)\right)^2\right)}$$

and phase:

$$\tan^{-1}\left(\sum_{j=0}^{q} b_j \sin(-\omega jT) / \sum_{j=0}^{q} b_j \cos(-\omega jT)\right)$$

This method may be used to provide the amplitude and phase response for any FIR filter. The transform is called the Fourier transform (defined in section 4), and it has a simple inverse. Conversely the filter coefficents may be obtained from the desired filter response using the same technique.

As a simple example, consider a low pass filter where the desired response, $H(\omega)$ is:

$$H(\omega) = \begin{cases} 1 & 0\omega \leq \omega_c \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$\begin{aligned} b_n &= \frac{1}{2\pi}\int_{-\pi}^{\pi} H(\omega)e^{in\omega}d\omega && (7) \\ &= \frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} e^{in\omega}d\omega && (8) \\ &= \frac{1}{2\pi}\left[\frac{1}{in}e^{in\omega}\right]_{-\omega_c}^{\omega_c} && (9) \\ &= \frac{1}{\pi}\sin(n\omega_c)/n && (10) \end{aligned}$$

But this means we need an infinite number of filter coefficients! True enough - real ``brick wall'' filters are impossible and sharp filters are hard to design. Some solutions:

- truncate: simple and effective if cycles are short
- window: use the Hamming window - minimises the power in the side-lobes
- Use a more complex filter design package, for example Parks-McClelland Remez Exchange algorithm, which designs optimal zero-phase FIR filters with arbitrary frequency responses.

Matlab implements all these, for example ``fir1(14, 0.5)'' is a 15 tap low pass filter that has a cutoff at half the maximum frequency. The filter coefficents are a windowed sinc funtion, plotted in figure 5 and the amplitude response is plotted in figure 6.

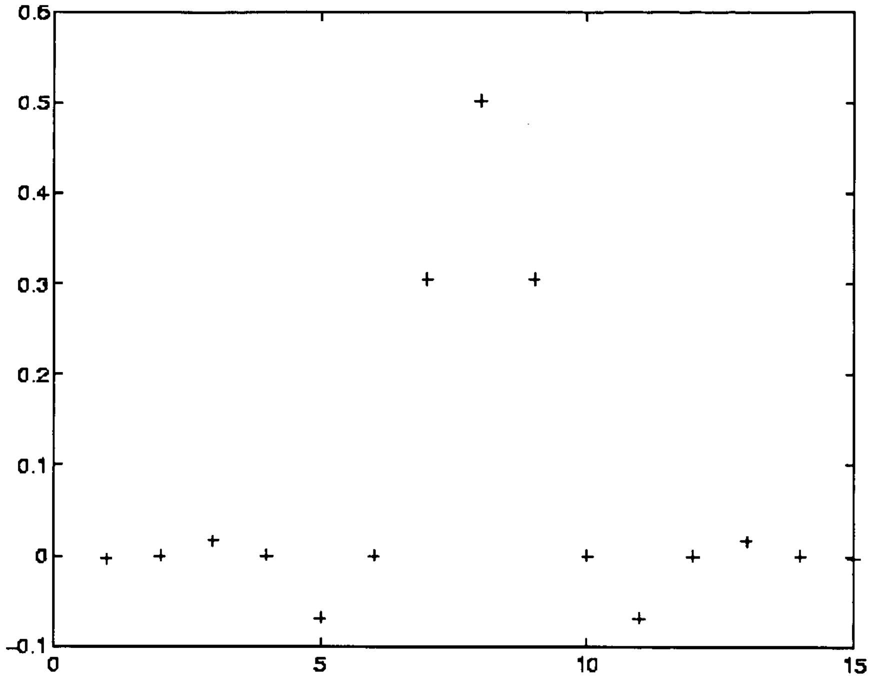

Figure 5: Example filter coefficiants: plot(fir1(14, 0.5), '+')

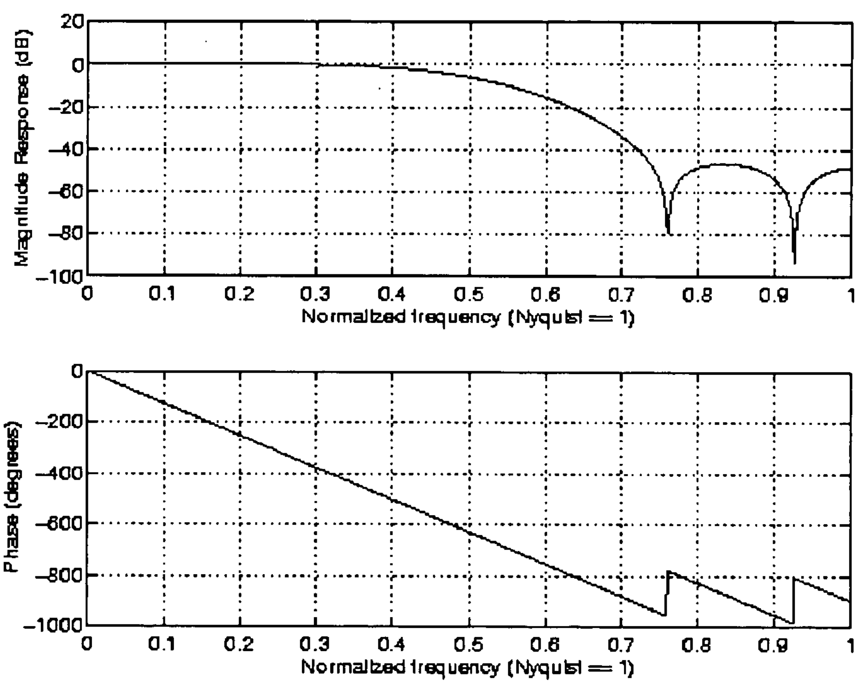

Figure 6: Example filter response: freqz(fir1(14, 0.5))

FIR filters are computationally expensive to implement but need not introduce phase distortions - useful in processing high quality speech.

IIR filters are often much more efficient, but can not be designed to have exact linear phase.

Filters and Fourier Transforms

NOTE:

Before reading these notes, see 15-462 "Basic Raster" notes:

http://www.cs.cmu.edu/afs/cs/academic/class/15462/web/notes/notes.html

OUTLINE:

- The Cost of Filtering
- Fourier Transforms

Properties of Convolution

both continuous and discrete convolution are

commutative: $a \otimes h = h \otimes a$

so the distinction between signal (a) and filter impulse response (h) is blurred

associative: $a \otimes (h_1 \otimes h_2) = (a \otimes h_1) \otimes h_2$

so instead of convolving signal a with complicated filter h, if h can be written as $h = h_1 \otimes h_2$, then a can be filtered in two passes: first with h_1 and then with h_2

Optimizing Filtering

- Filtering can be slow.
  - If the impulse response has a *support* (width of nonzero portion) of $S_x \times S_y$ pixels, then the cost of filtering is $O(S_x S_y)$ per output pixel. Filters with large support are expensive, in general. Filtering an $N \times N$ picture with an $S \times S$ impulse response costs $O(N^2 S^2)$ using standard formula - exorbitant!
- Certain filtering operations can be optimized.
  - Separable Filters
    - A filter h that can be written in the form $h[x,y]=h_x(x) \cdot h_y(y)$ is said to be *separable*. If the supports of h_x and h_y are S_x and S_y , then computing $a \otimes h$ directly costs $O(S_x S_y)$ per output pixel, but exploiting separability and associativity, we can do two-pass filtering, $(a \otimes h_x) \otimes h_y$, with cost of only $O(S_x + S_y)$.
  - Box Filters
    - A 1-D box filter of width S can be computed in O(1) time per output pixel.
    - A 2-D box filter of size $S \times S$ can be computed in O(1) time also.
- Fourier convolution:
  - optimizes general 2-D convolution to $O(N^2 \log N)$, as we will see later.

Fast Box Filtering

1-D box filtering can be done in O(1) time per output pixel

A 1-D box filter of width $S=2K+1$ is: $$b[x]=\frac{1}{S}\sum_{t=x-K}^{x+K}a[t]$$

With this formula, cost is O(S) -- slow for wide filters

But note that $b[x+1]-b[x] = (a[x+K+1]-a[x-K])/S$, so if we compute incrementally, adding in at the leading edge of the filter window, and subtracting out at the trailing edge, we get this fast algorithm:

```
initialize b
for x
    output b
    b += (a[x+K+1]-a[x-K])/S
```

2-D box filtering can also be done in constant time per output pixel

Do you see how to generalize the 1-D add-in/subtract-out trick to 2-D?

I know of three ways to do this. (This comes up again for texture mapping).

Cost of Filtering Algorithms

ALGORITHM	1-D signal length = N filter width = S	2-D picture size = N×N filter size = S×S
straightforward	$O(NS)$	$O(N^2S^2)$
box filter	$O(N)$	$O(N^2)$
separable filter	N.A.	$O(N^2S)$
Fourier convolution with FFT	$O(N\log N)$	$O(N^2\log N)$

Frequency Domain

We can visualize & analyze a signal or a filter in either the spatial domain or the frequency domain.

Spatial domain: x, distance (usually in pixels).

Frequency domain: can be measured with either:

ω, **angular frequency** in radians per unit distance, or

f, **rotational frequency** in cycles per unit distance. $\omega = 2\pi f$.

We'll use ω mostly.

The **period** of a signal, $T = 1/f = 2\pi/\omega$.

Examples:

The signal [0 1 0 1 0 1 ...] has frequency $f = .5$ (.5 cycles per sample).

The signal [0 0 1 1 0 0 1 1 ...] has frequency $f = .25$.

Fourier Transform

The **Fourier transform** is used to transform between the spatial domain and the frequency domain. A **transform pair** is symbolized with "$\leftrightarrow$", e.g. $f \leftrightarrow F$.

SPATIAL DOMAIN	$\leftrightarrow$	FREQUENCY DOMAIN
signal $f(x)$	$\leftrightarrow$	**spectrum** $F(\omega)$

Fourier Transform : $$F(\omega) = \int_{-\infty}^{+\infty} f(x) e^{-i\omega x} dx$$

Inverse Fourier Transform : $$f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega) e^{i\omega x} d\omega$$

where $i = \sqrt{-1}$. Note that F will be complex, in general.

Filtering Terminology

For a linear, shift-invariant filter, $\text{input signal} \longrightarrow \boxed{\text{FILTER}} \longrightarrow \text{output signal}$

A filter can be described in the spatial domain by its **impulse response**† $h(x)$, its response to a delta function input, as a function of position. Abbrev: IR.

$\delta(x) \rightarrow \text{FILTER} \rightarrow h(x)$

† a.k.a. **point spread function** in image processing

And it can be described in the frequency domain by its **frequency response** $H(\omega)$, its response to a sinusoid input as a function of frequency. Abbrev: FR.

$\sin(\omega x) \rightarrow \text{FILTER} \rightarrow \mathrm{H}(\omega)\sin(\omega x)$

$\mathrm{H}(\omega)$ is the **gain** of the filter at frequency ω.

The FR is the Fourier transform of the IR: $h(x) \leftrightarrow \mathrm{H}(\omega)$.

Note the terminology distinction. For a signal: signal↔spectrum,

but for a filter: impulse response↔frequency response.